

Comenzar un proyecto.

Introducción.

¿Cansado de leer libros que no te dicen nada, y solo hacen que repetir lo que pone en los menús?. ¡Pues eso se va a acabar!. Con el presente libro, “Un ejemplo para WinCC”, seguirás sin tener ni idea probablemente de lo que deseabas saber, pero te aseguro que por lo menos te lo pasarás bien.

Si estás harto de ver como los pdl son mas estáticos que los ojos de espinete, o que te queden mas horribles que Jesús Gil en bañador, descubre con este libro como realizar todas esas acciones que convierten tu scada en uno de los más admirados (por ti, claro).

Para trabajar con este libro se requiere disponer primeramente de:

- Conocimientos de WinCC básicos o medios¹,
- Tener instalado WinCC, y para algunos capítulos además Step 7.
- Tener un PC²

Si dispones de todo esto, y tiempo para realizar este proyecto, vamos para adelante como los de Alicante, que el ejemplo tiene tela.

¹ Si no dispones de estos conocimientos, ver mi libro anterior “Manual WinCC V4.02”

² ¿Qué qué es un PC?. Bien, este no es tu libro, deja de leer ahora mismo y saldrás ganando.

Lo primero: Definir la pantalla.

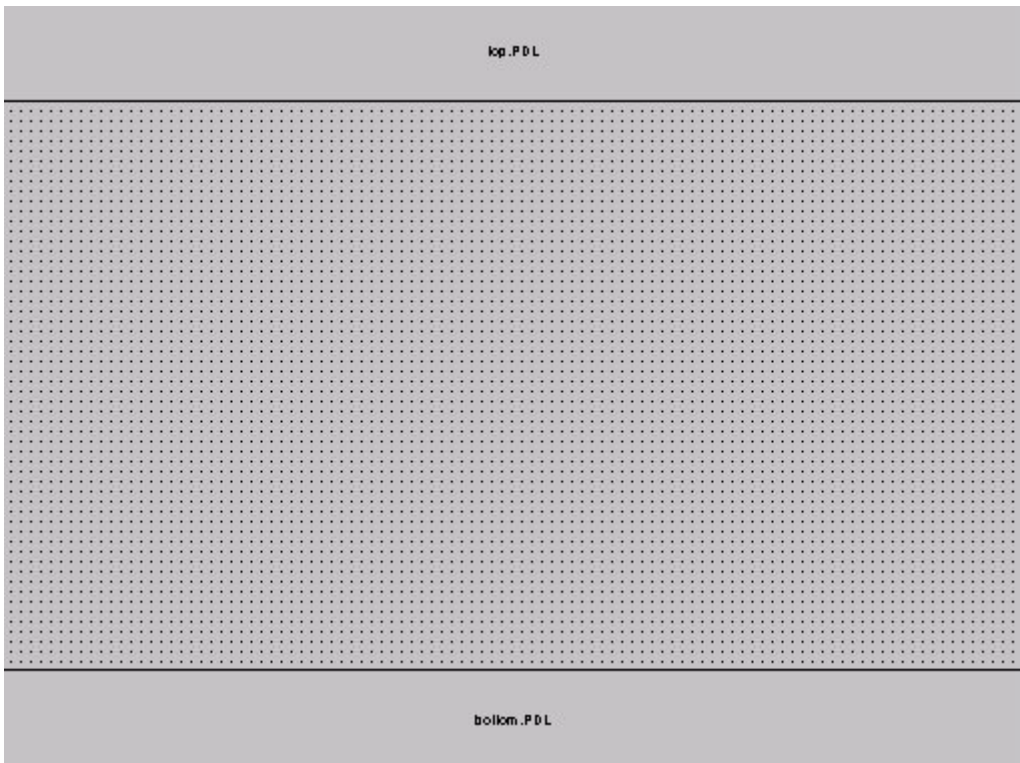
El primer paso en nuestro proyecto será definir la pantalla en la que vamos a trabajar. Pondremos en la parte superior una zona común de 1024x100, y en la zona inferior otra zona común de 1024x100. Como disponemos de una resolución de 1024x768, nos queda la zona central de 1024x568 para poner nuestra ventana actual.

Zona superior: para acceder a las diferentes pantallas del proyecto
Zona central: para mostrar la pantalla en la que estemos actualmente
Zona inferior: para acceder a los botones de funciones.

Como hemos dicho, se supone que sabes de WinCC, por lo que te lo diré en una sola frase: “créate un proyecto, la pdl inicial que se llame **start.pdl** , que tenga un tamaño de 1024x768, la guardas, te haces otra que llamas **top.pdl**, de tamaño 1024x100, y otra que se llame **bottom.pdl**, del mismo tamaño”.

Bien, teniendo esto claro, abrimos start.pdl, y en la parte superior colocamos una picture window de tamaño 1024x100, y seleccionamos en picture window->miscellaneous->picture name: top.pdl. También activamos aquí border: yes.

Realizamos la misma acción para la ventana de abajo, colocando otra picture window, pero esta vez en sus propiedades ponemos position y:668, y como picture name bottom.pdl. También ponemos el borde activo. Así, dispondremos de nuestra primera ventana, mas simple que el funcionamiento de un botijo, pero la base sobre la que vamos a trabajar. Debe tener el siguiente aspecto:



Como salir de WinCC.

Aún no hemos empezado y ya estamos saliendo, esto va bien. Para salir de WinCC nos ponemos un botón en bottom.pdl. La función para salir puede ser:

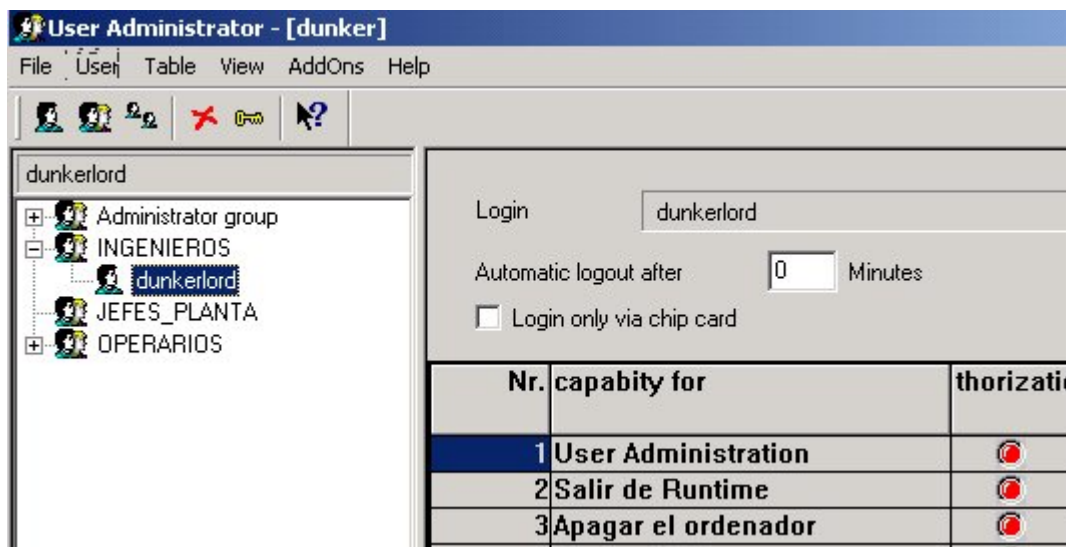
- DeactivateRTProject(); Sale del runtime de WinCC
- DMExitWinCCEx (DM_SDMODE_SYSTEM); sale de Windows y apaga el ordenador.
- DMExitWinCCEx (DM_SDMODE_REBOOT); sale de Windows y reinicia el ordenador.

Como este es un proyecto serio (jeje), pondremos en el botón de exit, que se encuentra en la biblioteca, y le asociamos la función DMExitWinCCEx (DM_SDMODE_SYSTEM).

Bien, ya salimos de WinCC, pero claro, cualquiera puede apagar el ordenador. Tendremos que controlar quien toca y quien no en nuestros ordenadores.

Controlando a los usuarios.

Vamos a organizarnos un grupo de usuarios, dentro del user administrator. Nos editamos los dos primeros privilegios, uno como “salir de wincc” y otro como “apagar el ordenador”.



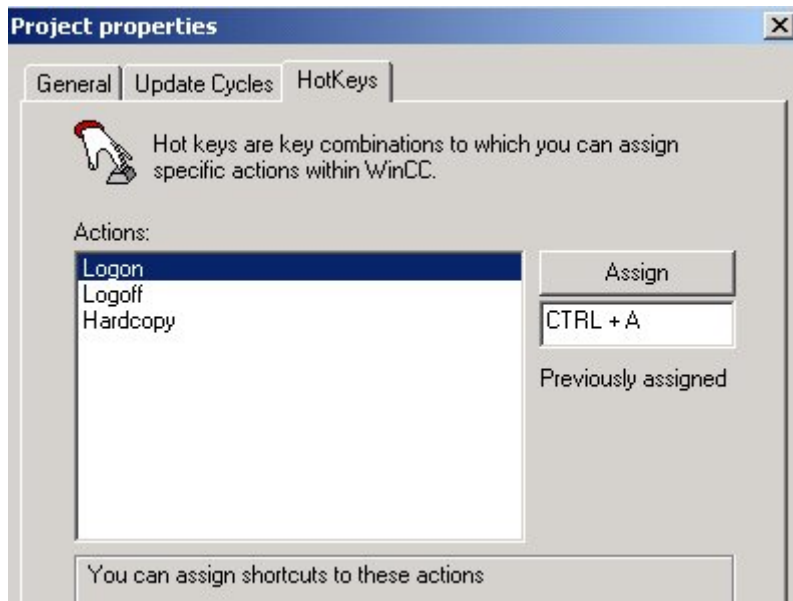
Les ponemos un automatic logout, para que si no se desactivan ellos, los desactive el sistema pasados x minutos.

Bien, ya tenemos nuestros usuarios, queda asignar el nivel de protección al botón de salir de la picture start.pdl. Volvemos a ella, y en la propiedad miscellaneous->authorization seleccionamos apagar el ordenador.

Ahora ya no podrá salir del wincc cualquiera, sino solo los que tengan el privilegio de apagar el ordenador, y cuando se logen y pulsen el botón se apagará el ordenador automáticamente.

Comenzar un proyecto

Pero: ¿cómo se logan³ los usuarios?. Hay dos maneras, una cutre salchichera y otra guay y complicada. Vamos con la cutre. En el control center, pinchando con el botón derecho sobre el nombre del proyecto, aparece la ventana en la que se puede asignar una combinación de teclas a login (logarse) y otra a logout (deslogarse)



Pero claro, se debe de recordar cual era esa combinación, y siempre aparece el listo que quiere un botón para activar a los usuarios. Ya la hemos cagao, pensarás (y estás en lo cierto), ya que la otra va a través de C. Quita las asignaciones que acabamos de hacer, que no vamos a hacerlo por ese camino, sino por el del botón. Nos colocaremos un botón en la picture bottom.pdl (recuerda, bottom será nuestra ventana para los botones de funciones, mientras que top será nuestra ventana para los botones de entrar a las diferentes partes en las que se muestra la instalación. El botón lo cogemos de la biblioteca:



Ahora viene lo bueno, el código C, asociado en el evento mouse actino:

```
#pragma code("useadmin.dll")
#include "PWRT_api.h"
#pragma code()

long pepe;
pepe=strcmp(GetTagCharWait("@CurrentUser"), "");
printf("pepe: %d", pepe);
if(pepe)
{
    PWRTLogout();
}
else
```

³ Logarse: darse de alta en un sistema protegido, introduciendo tu nombre y tu password.

Comenzar un proyecto

```
{  
PWRTLogin('c');  
}
```

Con esta función te logas y deslogas con el mismo botón (ha sido fácil, eh?).

Para acabar de rematar la faena, pondremos en top.pdl una I/O box de solo lectura, que va asociada a la variable @current_user. Esta variable la genera el sistema y es una cadena de texto que posee el texto del usuario que actualmente está logado. Con ello podremos saber en todo momento si estamos activados en el sistema o no.

La cosa va cogiendo color. Pero nos falta un detalle. Nosotros estamos poniendo los usuarios en el sistema desde el User administrator, y quitamos, ponemos y cambiamos passwords los que queremos y mas. Pero, ¿y cuando nos vayamos de la instalación?. ¿quién podrá cambiar los passwords, o añadir un usuario?. Tenemos que poner un segundo botón en la pantalla bottom.pdl. Ese botón nos accederá al user administrator. ¿Cómo?, fácil, así:

```
ProgramExecute("D:\\SIEMENS\\WINCC\\BIN\\PASSCS.EXE");
```

Asociando este código al botón de administración de usuarios, ya puede un usuario dar de alta y borrar, es decir, administrar a los demás usuarios. Claro está, a este botón creado, deberemos de asociarle un nivel de password, para que únicamente puedan entrar los usuarios con capacidad para borrar a otros usuarios.

Realizando nuestra primera picture

Ya estamos en condiciones de comenzar nuestra primera picture. Hasta ahora, todo lo hemos hecho en start.pdl. Vamos a utilizarla como plantilla para las demás pictures que generemos (también puede guardarla como plantilla.pdl, y a partir de entonces, en vez de hacer una nueva picture, abres la plantilla.pdl y la guardas con otro nombre.

La primera picture la llamaremos Tuberías.pdl, y lo primero que tendremos que hacer es llamarla desde nuestra ventana start.pdl. Lo haremos poniendo un botón en top.pdl, pero atención, si no llamamos a la picture tuberías mediante la función openpicture de C, sino a través de la ventana configuration dialog, no nos funcionará el salto, ya que estamos realizando la llamada desde un botón que está incrustado en una picture window. Por lo tanto, lo que haremos será poner nuestro botón, y en el evento mouse actino colocaremos la instrucción:

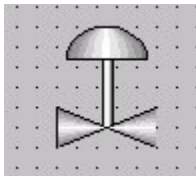
```
OpenPicture("tuberias.PDL"); //Return-Type :void
```

Ya tenemos el botón para entrar. Ahora necesitamos un botón para volver a la página principal. Pondremos en la picture top.pdl una imagen a la derecha con el logo de la empresa. Cuando se pinche en el logo, saltaremos a la página inicial con la instrucción:

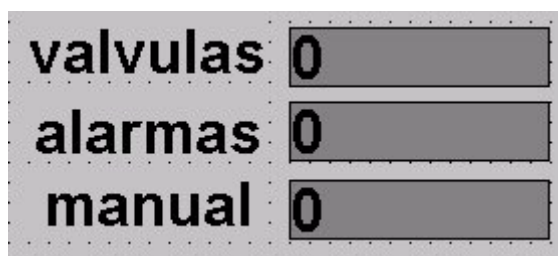
```
OpenPicture("start.PDL"); //Return-Type :void
```

Comenzar un proyecto

Comenzaremos con una simple válvula, que sacaremos de la biblioteca de WinCC. Con la versión 5.0 se ha añadido una librería muy interesante de objetos dentro de Simatic HMI symbols library 1.2. Dentro de la carpeta valves disponemos de una válvula como la figura:

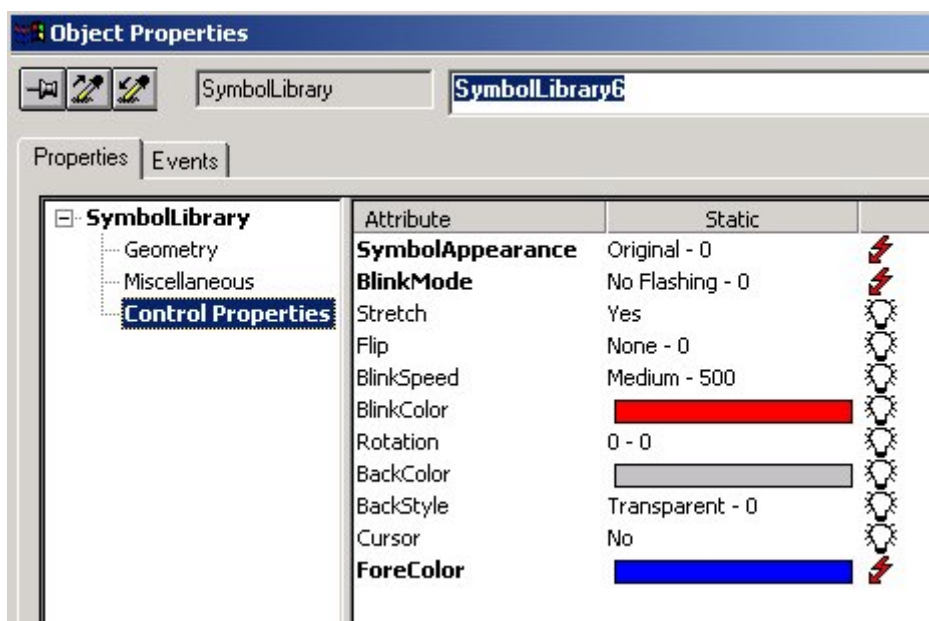


Vamos a generar tres variables, todas unsigned 32 bits. En la picture, colocamos tres I/O box asociadas a estas tres variables. La primera de ellas será estados_válvulas, y contendrá el estado actual (en marcha o parada) de 32 válvulas. La segunda será alarmas_válvulas, y la tercera manual_válvulas (manual válvulas indica si la válvula se encuentra gobernada en manual o en automático).



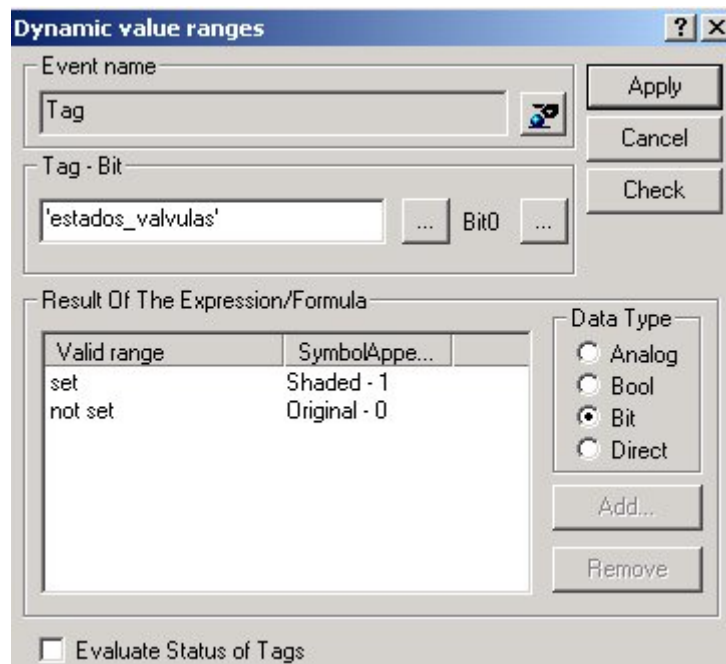
¿Qué buscamos?. Simplemente visualizar el estado de una válvula, pero sin utilizar una variable para cada estado, sino con 3 variables poder, consultando los bits de que consta la doble palabra, disponer de 32 .

Volvemos a las propiedades de la válvula que hemos colocado al principio desde la librería. La variable symbolappearance nos indicará si la válvula se encuentra en marcha o parada. La variable blinkmode nos indicará si tiene alarma, mientras que la de forecolor si está en manual o automático.

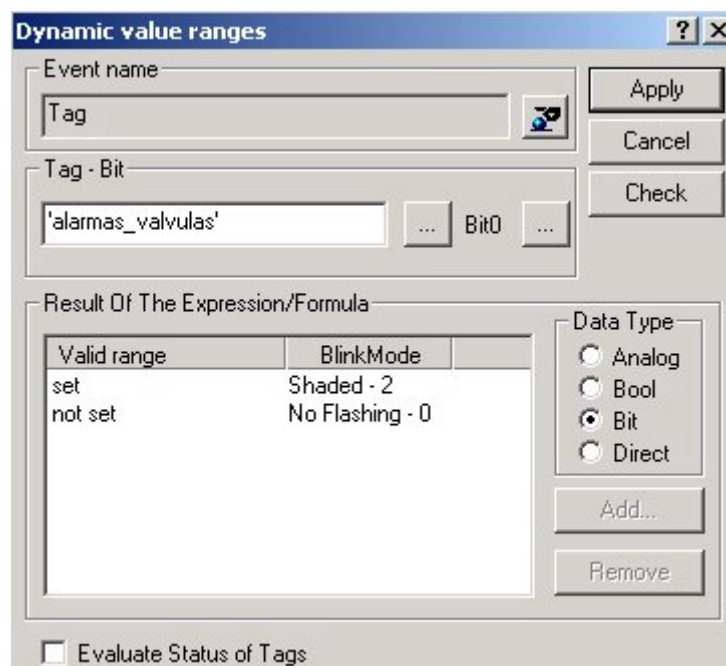


Comenzar un proyecto

Veamos que colocaremos en symbolappearance:

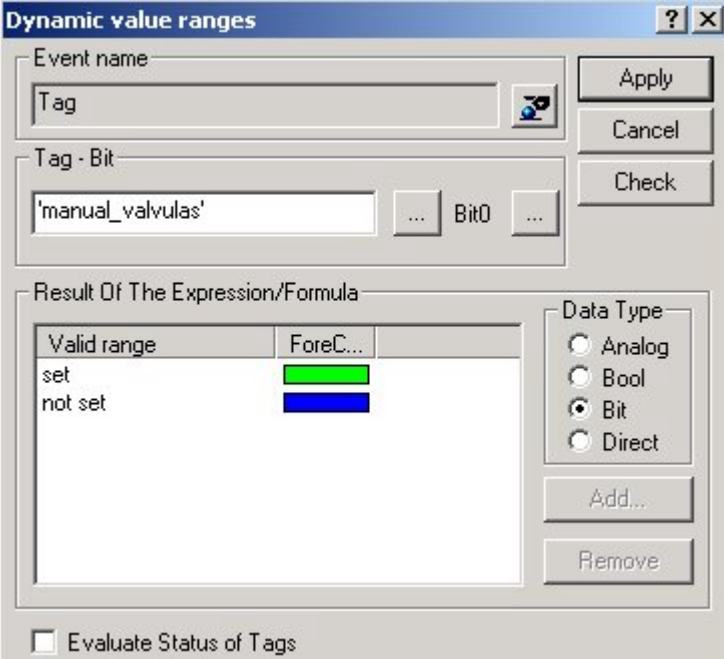


En la propiedad blinkmode realizaremos el siguiente dynamyc dialog:



Y por último, para la propiedad forecolor realizaremos un dynamyc dialog que nos identifique cuando la válvula está en manual y cuando en automático.

Comenzar un proyecto



The 'Dynamic value ranges' dialog box is used to configure the logic for a tag. It includes fields for the event name and tag-bit, a table for defining valid ranges with corresponding foreground colors, and options to select the data type (Analog, Bool, Bit, or Direct). Buttons for 'Apply', 'Cancel', 'Check', 'Add...', and 'Remove' are provided. A checkbox at the bottom allows for evaluating the status of tags.

Event name: Tag

Tag - Bit: 'manual_valvulas' ... Bit0 ...

Result Of The Expression/Formula

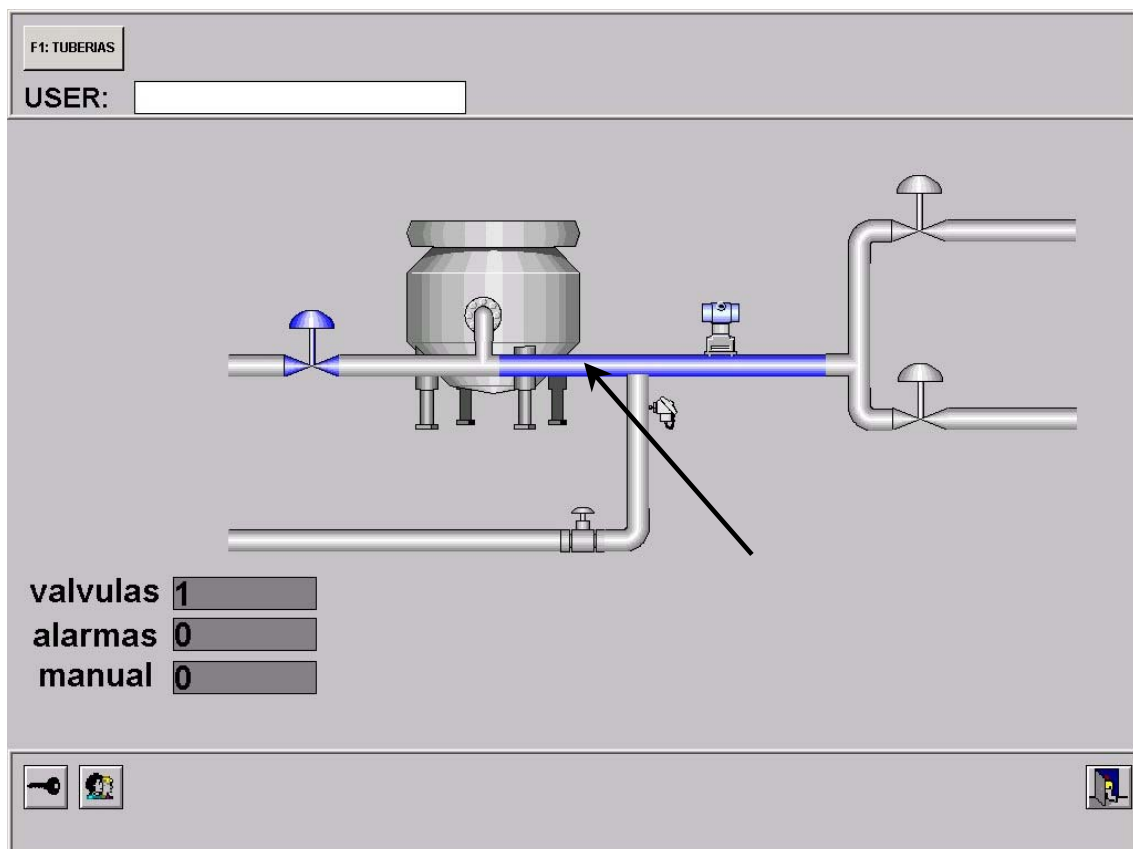
Valid range	ForeC...
set	[Green Box]
not set	[Blue Box]

Data Type:
☐ Analog
☐ Bool
☒ Bit
☐ Direct

Buttons: Apply, Cancel, Check, Add..., Remove

☐ Evaluate Status of Tags

Con esto tenemos solventado el tema de una válvula. Vamos a dibujarnos unas cuantas mas, junto con alguna que otra tubería (todas salen de la biblioteca anteriormente mencionada).



Comenzar un proyecto

El problema viene en el momento queremos colorear la tubería señalada con una flecha en la imagen. ¿Por qué?. Fácil, porque depende de tres válvulas. Por lo tanto no debe de colorearse cuando esté un bit, sino tres. Para ello, necesitaremos realizar una pequeña instrucción en C, que nos devolverá un valor entre 0 a 1.

En lugar de utilizar un dinamic dialog, en el evento symbolapperance asociamos una función C como la siguiente:

```
long valor1,valor2,valor3,resultado;  
double var1,var2,var3;  
  
var1=GetTagDWordWait("estados_valvulas");  
  
valor1=var1 && 2;  
valor2=var1 && 4;  
valor3=var1 && 8;  
  
if (valor1 | valor2 | valor3) resultado=1;  
else resultado=0;  
  
return(resultado);
```

Como podemos ver, esta función devuelve un 1 si se activa el bit 1, el 2 o el 3 de la doble palabra estados_válvulas. Aunque esta función se ejecuta correctamente, sería mejor no hacer este tipo de “cosas”⁴ en el scada, y utilizar una marca en el plc para las tuberías, con lo que no haría falta consultar esta función de C sino realizar una asignación como se hizo en la válvula inicial.

Bien con todo esto hemos aprendido:

- A entrar y salir del wincc de manera decente.
- A controlar quien se nos activa y quien no en el sistema, para poder echarle la culpa a alguien de que las cosas no van bien.
- A animar algo los objetos para que se muevan algo mas que epi y blas en una cama de velcro.

⁴ Léase cochinadas, marranadas y demás porquerías que se programan en un plc.

Saber en que picture estamos.

Mas adelante vamos a realizar muchas pantallas, a cual mas cutre, pero no por ello pantallas. Es muy interesante que sepamos en cual estamos en cada momento. Ahora es fácil, pero cuando el proyecto posee 50, recordar el nombre la que estamos en ese momento se vuelve tarea vergonzosa, porque siempre se oye la voz maliciosa “¿pero no lo habías hecho tu?” cuando buscas desesperadamente la picture en el graphics designer.

La solución está en poner el nombre de la picture actual en la pantalla, de tal manera que encontrarla luego para modificarla sea muy sencillo. Claro, esto además tiene la ventaja de que te corrige esa tendencia escatológica a llamar a las cosas cuando se está desarrollando caca o pedo o mierda, o cosas peores, que luego aflorar una vez el proyecto está acabado.

Como se hace esto es muy sencillo, siempre que lo hagas al principio del proyecto. Luego es mas complejo porque habrá que modificar todas las pictures.

Hay dos maneras, una fácil y mala y otra algo mas compleja y buena (como todo en la vida).

Vamos con la mala:

Abrimos la picture top.pdl. Colocamos a la derecha de la I/O de user_actual, un texto que en su propiedad font->text tendrá el siguiente código C:

```
return(GetParentPicture(lpszPictureName)); //Return-Type :char*
```

Lo que hacemos en realidad es asignarle a dicho texto no el nombre de la picture en la que está, que evidentemente siempre será top.pdl, sino el nombre de la ventana padre en la cual e aloja top.pdl, que según estemos en una será una, y en otra será otra.

Que fácil, ¿no?. Pues no. El problema radica en que esta comprobación del nombre de la ventana padre lo controlas, porque observa cuando hagas el código C que arriba tienes un trigger, que debes poner a uno o dos segundos. Pero es un poco tonto perder tiempo en chequear de manera cíclica si has cambiado de pantalla, sobre todo porque el que cambia eres tu, y dispones en le sistema de ese evento. Estamos cargando al sistema de una tarea que no es necesaria, ya que solo queremos que cambie el nombre en dicho texto cuando cambiemos de pantalla, no cada un segundo.

Vamos pues con la solución buena. Consiste en la picture plantilla colocarse en el evento open picture el siguiente código:

```
SetTagCharWait("PANTALLA_ACTUAL",lpszPictureName);
```

Con esto copiamos el nombre de la pantalla actual a una variable interna, que hemos definido previamente, pantalla_actual, y que es de tipo texto de 8 bits.

Ya solo tenemos que asociar en top.pdl al texto anterior, en la propiedad font->text la variable pantalla_actual con trigger upon change. Claro está, las pantallas que hemos hecho ya, como son start y planta deneremos de ponerle el mismo código en el evento open picture que ha plantilla. Pero a partir de ahora ya no será necesario, porque todas las pictures las vamos a sacar de plantilla.

A mi me gusta mas la segunda opción, pero tu mismo, te doy las dos.

n el siguiente capítulo vamos a hacer algo que nos sale muy bien: generar alarmas, pero esta vez no entre los que nos rodean, sino dentro del WinCC.